UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Advanced Center for Electrical and Electronic Engineering

# Short-term hydrothermal generation scheduling using a parallelized stochastic mixed-integer linear programming algorithm

Esteban Gil and Juan Araya

Departament of Electrical Engineering
Advanced Center of Electrical and Electronic Engineering (AC3E)

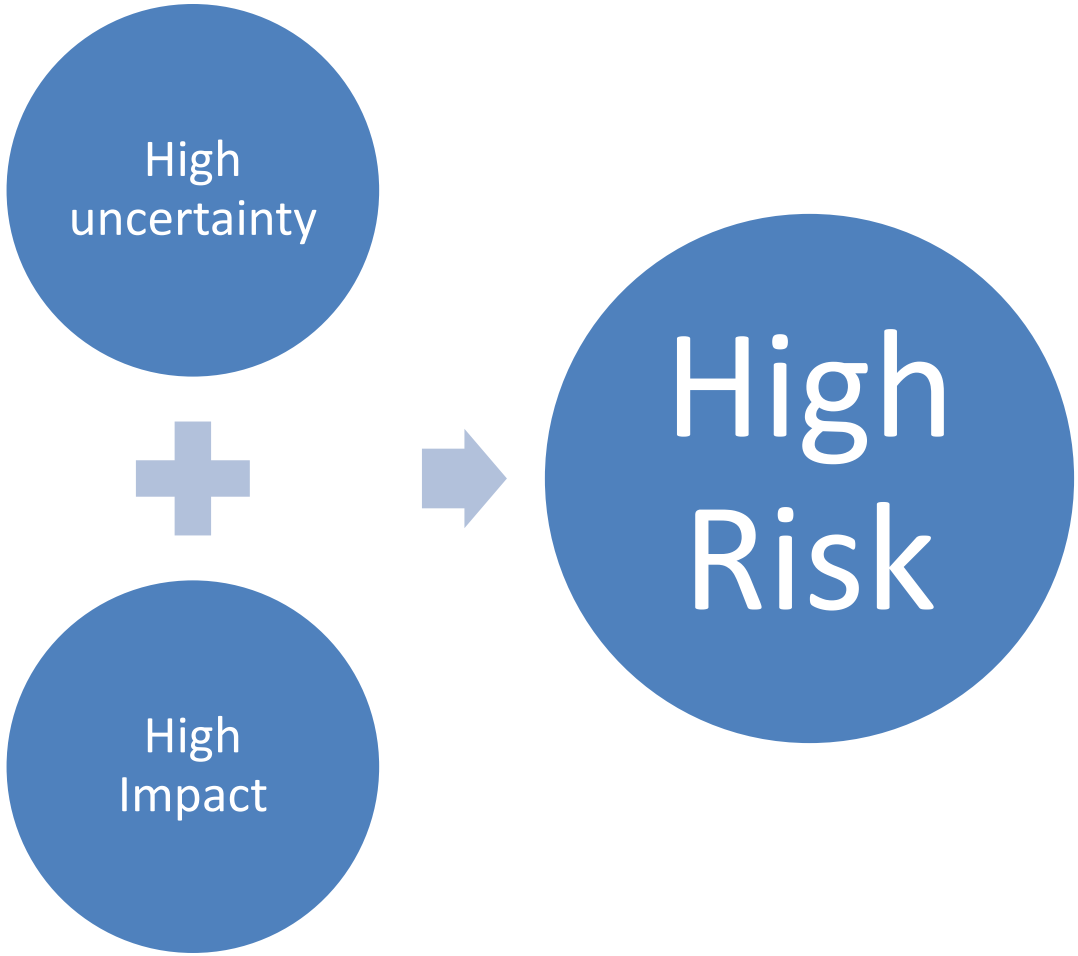
Universidad Técnica Federico Santa María (UTFSM)
Valparaíso, Chile

# Contents

# Decision making in power systems

Uncertainty:

- Demand
- Fuels
- Hidrology
- Prices
- Outages



# Chilean Central Interconnected System (SIC)

Generation

- More than 14 GW
- 54.6% thermal
- 43.2% hydro

Transmission

- 14355 km of lines above 66kV

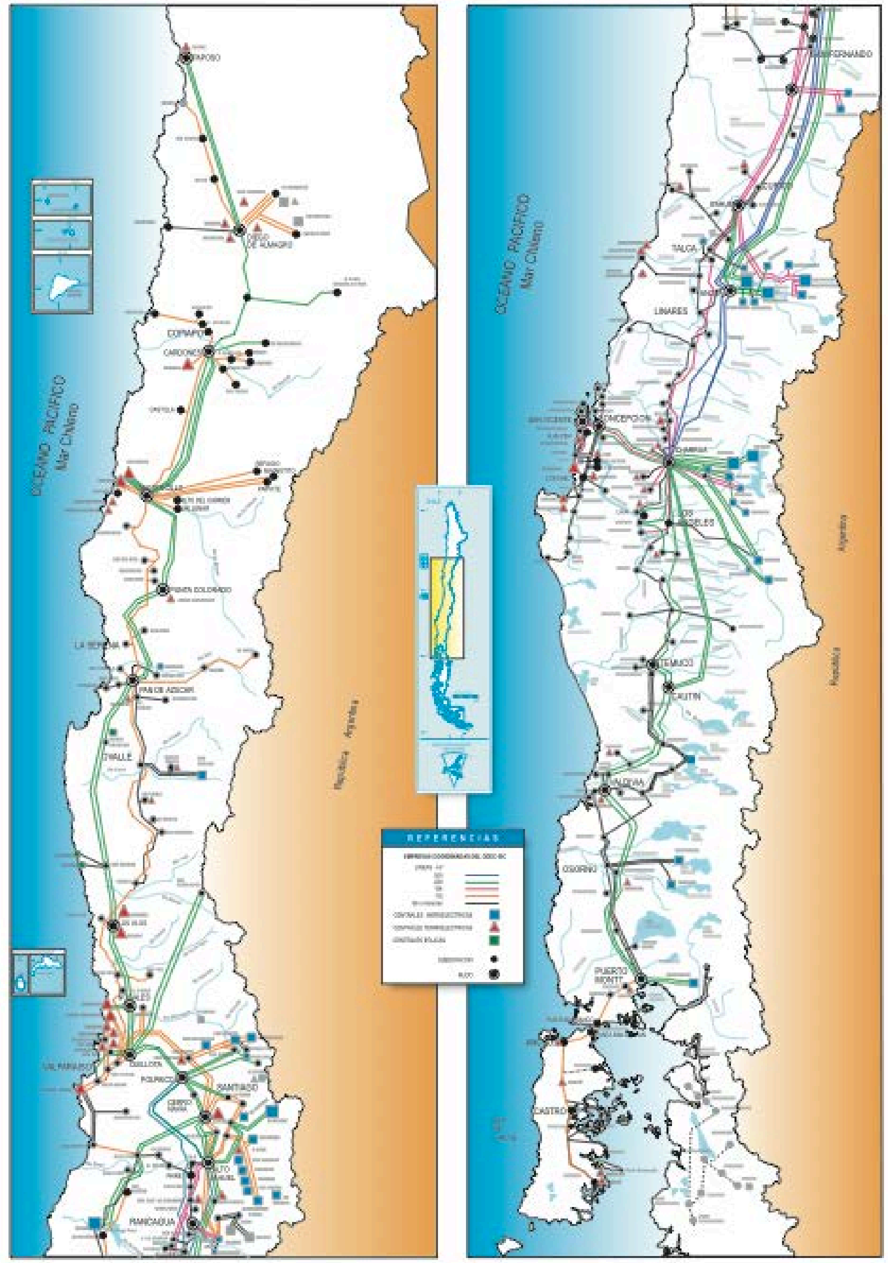
Demand

- 7282 MW maximum
- 92.2% of Chile's population

Operation

- Centrally dispatched
- Cost-based dispatch

SISTEMA INTERCONECTADO CENTRAL



# Short-term hydrothermal generation scheduling (STHTGS)

- Minimization of present operation costs plus future water costs

Minimize:

$$\sum_{t \in T} \{ y_t + \sum_{r \in R} FCF_r(Vol_{T,r}) + \sum_{n \in N} VoLL \cdot USE_{t,n} \}$$

$$y_t = \sum_{g \in G} \{ C_g^{op} \cdot P_{t,g} + C_g^{on} \cdot Y_{t,g} + C_g^{off} \cdot Z_{t,g} \}$$

- Subject to many constraints
  - e.g. demand satisfaction, water balance in reservoirs, DC-OPF, loading ramps, cascading hydro, and so on
- FCF comes from mid/long term models

# Short-term hydrothermal generation scheduling (STHTGS)

- Daily or weekly horizon, hourly resolution

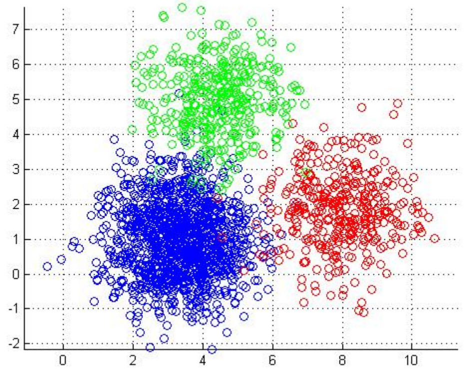
- 2 types of variables
  - First stage, grouped in $\mathbf{x}_s$
    - Unit commitments of thermal generators (integer)
    - Use of water in reservoirs (reservoir trajectories)
    - Need to decide now

  - Second stage, grouped in $\mathbf{y}_s$
    - Generation dispatch, flows in lines, and so on
    - Can decide once uncertainty unfolds

- Problem is MILP

# Stochastic programming

- Optimization under uncertainty
  - Uncertainty represented by *S* scenarios
    - Multivariate probability distributions represented by finite set of scenarios




  - Objective function: Expected value
  - Constraints must be satisfied for all scenarios

# Stochastic programming

- Deterministic
  - $|S|$ deterministic problems
  - $\mathbf{x}_s$ different for each scenario

$$\text{Minimize: } \{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s\} | \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s$$

- Stochastic
  - $|S|$ times larger than each deterministic problem

$$\text{Minimize: } \sum_{s \in S} \omega_s \{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s\} \mid \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s \ \forall s \in S$$

# Stochastic programming

- Stochastic problem

$$\text{Minimize: } \sum_{s \in S} \omega_s \{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s\} \mid \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s \ \forall s \in S$$

  – Each sub-problem is independent
  – We need to ensure that first stage variables are the same across scenarios

- Solution: Non-anticipativity constraints

$$\text{Minimize: } \sum_{s \in S} \omega_s \{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s\} \mid \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s \ \forall s \in S$$

$$\text{subject to } \mathbf{x}_i = \mathbf{x}_j \ \forall i, j \in S$$

  – Sub-problems coupled by non-anticipativity constraints => decomposition

# Problem decomposition

## Progressive hedging algorithm

$k \leftarrow 0$

**for** all scenarios $s \in S$ **do**

$\quad \mathbf{x}_s^{(0)} \leftarrow argmin_{x,y}\{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s\} \mid \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s$

$\quad \mathbf{w}_s^{(0)} = \mathbf{0}$

**end for**

→ Independent solutions

$\overline{\mathbf{x}}^{(0)} = \sum_{s \in S} \omega_s \cdot \mathbf{x}_s^{(0)}$

→ Initial non-anticipative solution

**repeat**

$\quad k \leftarrow k + 1$

$\quad$ **for** all scenarios $s \in S$ **do**

$\quad\quad \mathbf{w}_s^{(k)} = \mathbf{w}_s^{(k-1)} + \rho(\mathbf{x}^{(k-1)} - \overline{\mathbf{x}}^{(k-1)})$

$\quad\quad \mathbf{x}_s^{(k)} \leftarrow argmin_{x,y}\{\mathbf{c}_x \cdot \mathbf{x}_s + \mathbf{c}_y \cdot \mathbf{y}_s + \mathbf{w}_s^{(k)} \cdot \mathbf{x}_s + 0.5\rho \|(\mathbf{x}_s - \overline{\mathbf{x}}^{(k-1)})\|\} \mid \mathbf{x}_s, \mathbf{y}_s \in \mathbf{Q}_s$

$\quad$ **end for**

→ Solve problem with augmented lagrangian

$\quad \overline{\mathbf{x}}^{(k)} = \sum_{s \in S} \omega_s \cdot \mathbf{x}_s^{(k)}$

→ Update non-anticipative solution

**until** $\|(\mathbf{x}_s^{(k)} - \overline{\mathbf{x}}^{(k)})\| \leq tolerance, \ \forall s \in S$

→ Convergence criterion

# Progressive hedging

- Most steps can be parallelized
  - Except calculation of non-anticipative candidate solution

- Decomposition by scenario
  - Similarly sized sub-problems => Similar solution times
  - Although sometimes differences in MIP solution times

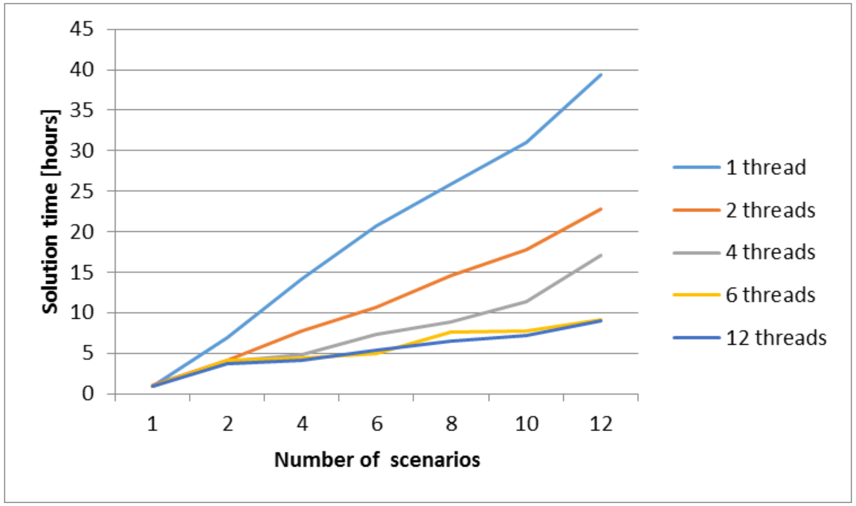
# Parallelization

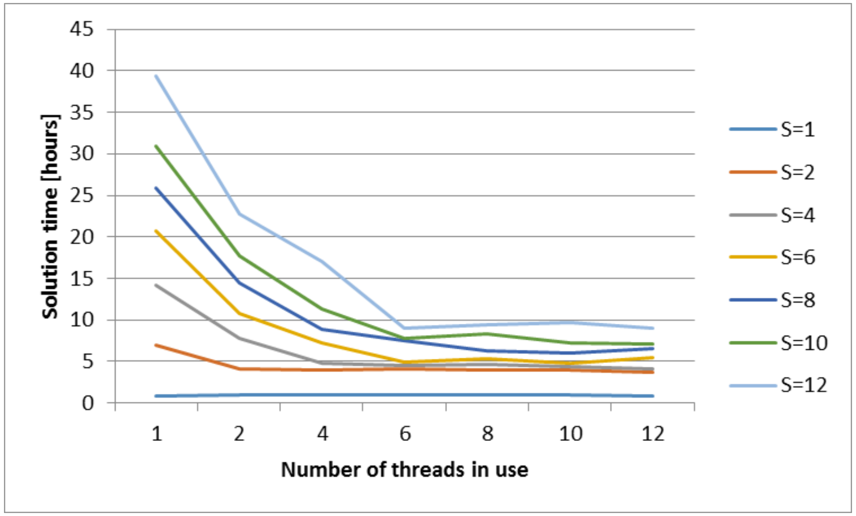
- Implemented in Fortran 95
  - We are migrating to Pyomo
- Solution of each sub-problem obtained with CPLEX
- Parallelization using hybrid OpenMP and MPI model
  - OpenMP inside each node
  - MPI protocol through Open MPI between the nodes
  - Shared memory architecture
    - Using a single node
    - All processes have access to the same physical memory
  - Distributed memory architecture
    - Using both nodes of the cluster
    - Network communications used to access memory on the nodes

# Simulation and problem size

- Computational experiments in two nodes of a cluster
  - Each node has 2 Intel E5 Xeon processors with 6 cores each
  - Each of the two nodes has 12 cores available.
  - CPLEX uses 2 cores per thread, so each node can run up to 6 CPLEX threads simultaneously
- Problem size
  - 152 buses, 202 transmission lines, 330 generators (205 thermal, 11 hydro with significant storage)
  - Weekly horizon, hourly resolution
  - Each sub-problem has 24691 rows, 367786 columns, with 962716 non-zero coefficients

# Shared memory results




- Solution time grows linearly with $|S|$
- Solution time decreases with more parallel threads
- Improvement stops when there are more threads than available cores
- To maintain decreasing trend, necessary to use more cores
  - Distributed memory

# Distributed memory results

| Number of scenarios | Simulation time | | Performance gain |
|---|---|---|---|
| | Shared memory | Distributed memory | |
| S=4 | 4.10 | 4.17 | 1.02 |
| S=6 | 4.35 | 5.40 | 1.24 |
| S=8 | 4.43 | 6.52 | 1.47 |
| S=10 | 4.67 | 7.15 | 1.53 |
| S=12 | 5.05 | 8.97 | 1.78 |

- Faster than shared memory
- Despite having twice as many cores available, it is not twice as fast as the shared memory scheme
- Parallel overhead, i.e. time required to coordinate parallel tasks

# Conclusions

- Stochastic Short-Term Hydrothermal Generation Scheduling formulated as SMILP
- In real-sized systems, stochastic problems cannot be solved without decomposition
- PH showed good performance:
  - Few iterations needed for convergence
  - Sub-problems of similar size
  - Good opportunities for parallelism and HPC

# Conclusions

- Two parallelization schemes
  - Shared memory using OpenMP
  - Distributed memory using hybrid OpenMP/MPI
- With shared memory
  - Scalability is limited by the number of cores
- With distributed memory
  - Can keep scaling up, but twice the number of cores does not mean twice the speed
  - Some performance loss due to parallel overhead

# Conclusions

- With more integer variables variability of solution times may increase
  - Difficulty to parallelize
  - Solution 1: Asynchronic resolution of sub-problems
  - Solution 2: Scenario grouping

# Conclusions and further work

- Application to other optimization problems:
  - Unit commitment
  - Expected profit maximization
  - Portfolio optimization
  - Mid- and long-term reservoir management
  - Investment
- Our newest results suggest that:
  - Value of the stochastic solution between 1% and 3%
  - As more scenarios are used, the marginal benefits of a more accurate uncertainty representation keep decreasing

# Questions?